TRAC: An Architecture for Real-Time Dissemination of Vehicular Traffic Information

Shravan Rayanchu
Sulabh Agarwal
Arunesh Mishra
Suman Banerjee
Samrat Ganguly

Technical Report #1604

August 2007

TRAC: An Architecture for Real-Time Dissemination of Vehicular Traffic Information

Shravan Rayanchu, Sulabh Agarwal, Arunesh Mishra, Suman Banerjee
Department of Computer Science, University of Wisconsin Madison, USA
Email: [shravan, sulabh, arunesh, suman]@cs.wisc.edu

Samrat Ganguly
NEC Laboratories, NJ, USA
Email: samrat@nec-labs.com

***Abstract*— Disseminating traffic related information to vehicular users is increasingly becoming a necessity nowadays in the context of Intelligent Transportation Systems (ITS). However, till date there does not exist an efficient model for information delivery to achieve this objective.**

We introduce TRAC, an architecture designed to support dissemination of location-sensitive real-time traffic information specifically targeted towards highly mobile users (vehicles moving at high speeds). We believe that this is the first in-depth study investigating the various components involved in designing a traffic information system using the publish/subscribe model. Additionally, we introduce the concept of virtual publishers and subscription predictions to make the traditional publish/subscribe paradigm better suited for applications like traffic information accumulation. Using extensive simulations, we argue for the feasibility of our proposed architecture.

I. Introduction

Traffic congestion is a major problem in many locations throughout the world. People lose valuable time, fuel and sanity when traveling at sub-optimal speeds. Over the past few years, travelers in the USA alone have spent over 3.7 billion hours in traffic delays [1]. The situation is getting worse because U.S. urban areas have not added the road capacity needed to keep up with the increase in demand [2]. Moreover, one cannot predict unexpected delays due to accidents, weather, etc. Since we cannot guarantee free-flowing traffic conditions we have to go to the next best step of providing accurate updates to allow users to make informed decisions based on current conditions to reach their destinations faster. A real-time information dissemination system announcing the current traffic conditions to the travelers is predicted to alleviate the situation to a large extent [1]. Consequently, recent efforts have focused on developing customized Intelligent Transportation System (ITS) services for vehicles [1], [11]. These services would utilize a variety of static and dynamic data from participating vehicles and the deployed road side infrastructure. The recently adopted Dedicated Short-Range Communication (DSRC) spectrum for vehicle-to-vehicle and vehicle-to-infrastructure multi-channel communication is expected to play an important role in realizing such a system supporting real-time information dissemination. Information could also be aggregated and disseminated using WiFi communities like FON [3]. This will alleviate the need of explicit deployment of access points.

[1]http://www.usatoday.com/news/nation/2005-05-09-traffic-study_x.htm
[2]http://www.npr.org/templates/story/story.php?storyId=4645279
[3]http://en.fon.com

Given that an infrastructure is made available, it is still not trivial to design a system that distributes on-demand information describing the traffic conditions and the available facilities in a geographic region. Earlier work [11], [14], [5], [3], [10] has proposed the use of publish/subscribe model to facilitate the dissemination of information in a mobile environment. However, as described in the next section, traditional pub/sub systems have their own limitations in being able to support traffic information dissemination. Issues such as scalability, location-dependence, intermittent connectivity and high mobility are not handled efficiently by traditional systems. Any feasible solution needs a communication architecture to take into account that the queries are *location sensitive* and that the sources for a particular query also tend to be *localized*. There is thus a need for fast and dynamic organization of the communication leveraging the crucial role of geographic position of the nodes and the locality properties exhibited by them.

To this end, in this study we propose an architecture for dissemination of time-sensitive data in mobile environments using the publish/subscribe model. Specifically, we focus on the problem of providing services that distribute real-time, on-demand information about traffic conditions (e.g. average speeds on a set of roads). We have named our system *Traffic Accumulator (TRAC)*.

Our key contributions are (1) We are the first to conduct an in-depth design and analysis of a traffic information dissemination system using a pub/sub model, (2) We define the TRAC architecture and abstractions such as *virtual publisher* and *subscription predictions* to make the traditional pub/sub system better suited for location sensitive applications in highly mobile environments and (3) We evaluate our proposed system through extensive simulations that accurately model vehicular movements in a real street map.

Problem Formulation

A number of recent research efforts have argued that publish/subscribe paradigm is suited for a variety of *mobile* information dissemination applications ranging from those dealing with safety critical information to commercial applications such as weather and traffic information dissemination [11], [14], [5], [3], [10]. However, our evaluation of the problem during the design of TRAC showed that traditional pub/sub paradigm is not directly applicable for the purposes of traffic information dissemination.

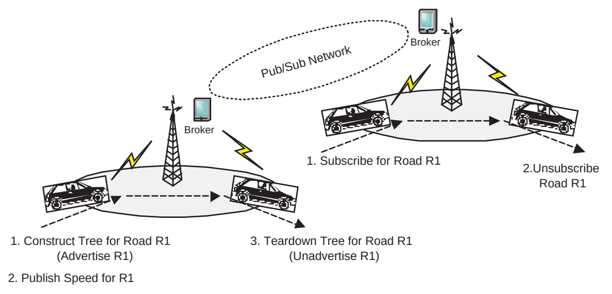

Fig. 1. Problem Scenario

Consider the above problem scenario (Figure 1) in which the DSRC access points are either explicitly deployed or access points are used from a WiFi community. In a pub/sub system, publishers post messages and subscribers register subscriptions with a software entity known as brokers. In our scenario, the access points can act as brokers that determine where and how the information should propagate. The vehicles would act as publishers as well as subscribers. Whenever the vehicles come under the range of an access point, they connect to the broker and participate in the pub/sub system. In the role of a publisher, a vehicle would send out advertisements about the road segments it has been through. These advertisements are flooded through the broker network and interested subscribers respond to them along the reverse path. This reverse path results in forming a broker multicast tree to ensure efficient propagation of published information to all interested subscribers. The vehicle would publish data (it would have sent out new advertisements as it traverses new road segments) as long as it is under the coverage of the access point and this data is forwarded along the multicast tree. When the vehicle moves out, unadvertisements would be propagated tearing down its multicast tree (steps shown in Figure 2).

If we were to implement a traffic accumulation and dissemination solution using the system described in [14], as a subscriber, the vehicle would request information about a path (a set of road segments) or a region that the user of the vehicle intends to traverse. If a broker residing on the access point has previously received any matching advertisements, it would send out the subscriptions along the reverse path thereby joining the corresponding multicast tree(s). Whenever the broker receives any matching publications for this subscription, it would forward them to the vehicle. The vehicle would continue to receive new publications as long as it is under the coverage of the access point. Unsubscriptions would be propagated, tearing down the corresponding branch(es) of multicast tree(s) when the vehicle moves out.

We now present the important requirements and characteristics of such a system and argue that traditional pub/sub systems are inadequate to satisfy them:

Number of Publishers vs. Subscribers: Traditional publish/subscribe systems assume that the number of subscribers typically outnumber the publishers. It is this assumption that justifies flooding of advertisements by *each publisher*. However, in our system, the number of publishers and subscribers are *large* and *equal* in number.

Connection time: In our problem scenario, *intermittent connectivity* is a rule rather than an exception. For vehicles moving at high speeds, the effective average connection time would be no more than a few seconds. Given the *connect-publish-disconnect* nature of the publishers in our system, the overhead of constructing a multicast tree for each publisher would far outweigh any savings achieved in the delivery of publications.

Location dependence: Since the data published by a particular vehicle changes with the publisher's location, the set of interested subscribers would also change. Previous schemes [14], [3] proposed to help traditional pub/sub systems cope with mobility are thus no longer applicable.

Locality of publications: We can expect the data published by vehicles under the range of an access point to be similar in content. Exploiting this locality of publications by reusing the multicast trees would reduce the cost of mobility, but traditional pub/sub system would be oblivious to such a property.

Subscriber locality: Subscribers would typically tend to be localized to a set of brokers rather than being interested in information coming from brokers spread throughout the network. The publishers and the subscribers interested in them would also tend to be geographically localized. It is important for our system to take advantage of these properties.

Significance of each publication: Unlike most pub/sub systems, supporting disconnected operation of subscribers in our system does not require a broker to store and replay the missed publications when a subscriber reconnects to the network. This is because the subscribers are not interested in the publications from any particular publisher (vehicle), but the *aggregate/average value*. Consequently, the importance of each published information diminishes.

Time-sensitivity of data: Traffic conditions can change very quickly and as a result the published information has to be delivered to the interested subscribers in *real-time*.

Mobility: Both *publishers and subscribers are mobile* in our problem which makes the system highly inefficient under traditional pub/sub system because of maintaining state per publisher and the cost incurred in building and tearing down trees.

Scalability: The architecture needs to be scalable as the number of participating users could be huge.

TRAC efficiently handles the above stated inadequacies and requirements using a novel concept of *virtual publisher* and optimizations like *subscription predictions*. Our objective while designing TRAC has been to distribute on-demand traffic information to highly mobile users at a minimal latency (multicast tree building time) while incurring a relatively low message overhead.

Roadmap

The rest of this paper is structured as follows. In the next section we present the design of the TRAC architecture. In Section III we present evaluation results from our experiments.

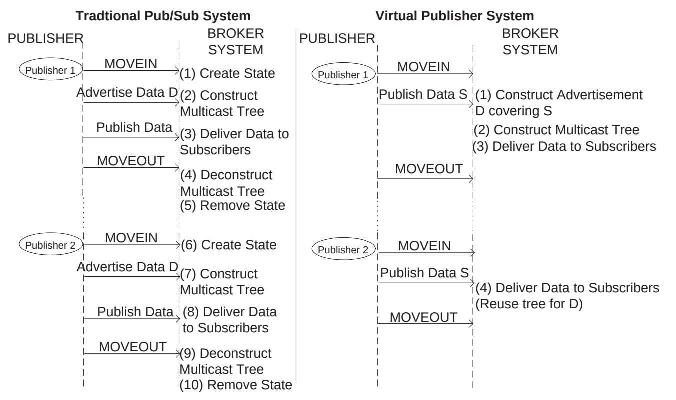
Tradtional Pub/Sub System
PUBLISHER
BROKER SYSTEM
Publisher 1
MOVEIN
(1) Create State
Advertise Data D
(2) Construct Multicast Tree
Publish Data
(3) Deliver Data to Subscribers
MOVEOUT
(4) Deconstruct Multicast Tree
(5) Remove State
Publisher 2
MOVEIN
(6) Create State
Advertise Data D
(7) Construct Multicast Tree
Publish Data
(8) Deliver Data to Subscribers
MOVEOUT
(9) Deconstruct Multicast Tree
(10) Remove State
Virtual Publisher System
PUBLISHER
BROKER SYSTEM
Publisher 1
MOVEIN
Publish Data S
(1) Construct Advertisement D covering S
(2) Construct Multicast Tree
(3) Deliver Data to Subscribers
MOVEOUT
Publisher 2
MOVEIN
Publish Data S
(4) Deliver Data to Subscribers (Reuse tree for D)
MOVEOUT

Fig. 2. Traditional Publisher vs. Virtual Publisher

In Section IV we highlight the prior work in this area of interest and finally we conclude in Section V.

II. TRAC: Architecture and Algorithms

In this section we discuss in detail the various components involved in TRAC. We also discuss the rationale behind our design choices.

Key Concept: Virtual Publisher

The primary drawback of traditional pub/sub systems is that they store state per publisher. Even when multiple publishers (vehicles) publish similar information, the system constructs a tree for each publisher. Although optimizations like *covering* [14] do help in reuse of the existing trees, they might not be effective in a highly mobile environment where connection times are very short. We propose using a *stateless* method of publishing. Instead of building an advertisement/multicast tree for each publisher, we introduce an abstraction called **Virtual Publisher**. The motivation lies in the fact that in location-sensitive applications, many publishers are publishing similar information and a subscriber is not interested in any particular publisher's data, but in the aggregated information. Thus, it is better for a system to construct a *tree per data item* rather than a tree per publisher.

When a publisher connects to a broker, it simply publishes the data and disconnects. The broker then sends out covering advertisements for the data to other brokers in the network. When the subscriptions from the interested subscribers reach the broker, a multicast tree for the data is built. The broker then publishes the data. However, if a tree for the data already exists in the system, a new branch created at the broker grafts onto it opportunistically and the existing tree is reused (Figure 2). This implies that the broker is now acting as a publisher on behalf of the actual publisher. The broker is the proposed virtual publisher of data.

Virtual publisher results in reduced advertisement cost, as the trees are not built for every publisher. The unadvertisement costs are saved to a large extent as the advertisement/multicast tree is not torn down when the publisher moves out. However, if no new publishers attach to the broker for an extended period of time, brokers send out the unadvertisements tearing down

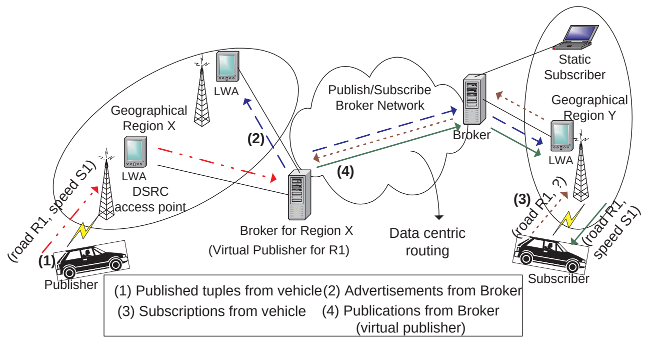

Fig. 3. TRAC: Architecture

the virtual publisher tree as this would reduce the unnecessary propagation of subscriptions towards the broker. It is worth noting that in TRAC, such cases are few and far between as the number of publishers are large and the locality of their publications is very high.

Entities in TRAC

We now describe the entities that participate in TRAC. Vehicles act as publishers as well as subscribers. A *Light weight agent (LWA)* resides on the DSRC/802.11 access points to which the publishers/subscribers connect and participate in the publish/subscribe operation. An LWA interacts with the publishers, subscribers and a broker. The broker module may reside on the access point itself, or might be located on a separate node. It is responsible for content-based routing and interacts with other brokers in the pub/sub system.

Publisher/Subscriber: In TRAC, real-time traffic information has to be collected before it can be disseminated to the interested subscribers. We propose to collect this data directly from the vehicles. TRAC assumes that the vehicles are equipped with GPS receivers to record accurate positioning information. Vehicles might also have an optional on-board diagnostics system (OBD) interface that can be used to collect data (speed of the vehicle, acceleration, etc.) from the electrical/mechanical sensors installed on the vehicles. Vehicles should also be equipped with a wireless network interface (802.11 or DSRC) to communicate with the access points. TRAC also assumes that vehicles come installed with a navigation module that provides a simple abstraction of the road network [2] and has the capability of mapping GPS coordinates (latitude, longitude) to road segments. Each vehicle periodically records the following information using GPS/OBD: (1) Timestamp of the observation (2) Location of the vehicle, in terms of latitude and longitude (3) Speed, in mph (4) Heading, indicating the direction of the vehicle's motion. These readings are fed into the navigation module that stores them. Whenever a vehicle comes under the range of an access point, it connects to the LWA residing on the access point and publishes the stored information since the last publication. These operations are transparent to the user. The user can query for information about a set of roads, or can request customized traffic alerts through the service layer.

The subscriptions are stored with the vehicle and whenever the vehicle comes under the range of an access point, it connects to the LWA and forwards the subscriptions to it. Upon receiving the publications, appropriate notifications (for eg. an update of the speeds on the roads the user had requested) are presented to the user through the service layer.

Procedure Set 1 : Basic Operations of LWA

```
Procedure ReceiveFromUser(Subscription $s$)
  $subscriptionTbl \leftarrow (s.sender, s) \cup subscriptionTbl$
  Forward $s$ to Broker
  if Publication $p$ in Cache matches $s$ then
    Send $p$ to $s.sender$
Procedure ReceiveFromBroker(Publication Set $P$)
  Update Cache with $P$
  for each Publication $p \in P$ do
    if $s \in subscriptionTbl$ matches $p$ then
      Send $p$ to $s.sender$
```

Light Weight Agent (LWA): The Light Weight Agent (LWA) would reside on the DSRC/802.11 access points explicitly deployed by the DOT or they could run on rented hot-spots providing the service. We intend to keep the operations of LWA very simple. An LWA interacts with the vehicles (publishers and subscribers) and the broker module it is connected to. An LWA comprises of a connection table and subscription table. The memory requirements would be minimal as the number of subscribers connected to an LWA at any point of time would not be many. The LWA also maintains a cache of publications which is purged periodically. As in [14], [5], we assume that the user (vehicle) mobility is based on MOVEIN and MOVEOUT operations. These operations offer users the ability to reconnect to and disconnect from the system respectively. When the LWA receives a MOVEIN from the user, it creates a state in its connection table. The user would then send publications to the LWA. The LWA passes the publication set on to the broker module to which it is connected. Other operations of LWA are presented in Procedure Set 1.

Broker Module: The basic operations of a broker in TRAC are similar to that of a traditional pub/sub system, i.e. the broker is responsible for routing of advertisements and publications, subscription matching and building the multicast trees. In TRAC, the broker also acts as a virtual publisher by periodically publishing the aggregate information received from the LWAs in its region. Additionally, it participates in optimizations for subscriber mobility. The details of the operations of broker module are presented in Procedure Set 2.

Procedure Set 2 : Basic Operations of a Broker

```
Procedure ReceiveFromLWA(Publication Set $P$)
  for each Publication $p \in P$ do
    $advSet = \phi$
    if $p \in pubTbl$ then
      Update $p.speed$
    else
      $pubTbl \leftarrow p \cup pubTbl$
      $advSet \leftarrow p \cup advSet$
  Construct Advertisement $a$ such that $\forall p \in advSet$, $a$ covers $p$
  Send Advertisement $a$ to all neighboring brokers
Procedure ReceiveFromBroker(Advertisement $a$)
  $inAdvTbl \leftarrow a \cup inAdvTbl$
  for each $s \in inSubTbl$ matching $a$ where $s.sender \neq a.sender$ do
    if $(s', n) \notin outSubTbl$ such that $s'$ covers $s$ and $n = a.sender$ then
      $outSubTbl \leftarrow (s, a.sender) \cup outSubTbl$
      Send $s$ to $a.sender$
  for each neighbor $n$ where $n \neq a.sender$ do
    if $(a', n) \notin outAdvTbl$ such that $a'$ covers $a$ then
      $outAdvTbl \leftarrow (a, n) \cup outAdvTbl$
      Send $a$ to $n$
Procedure ReceiveFromBroker(Subscription $s$)
  $inSubTbl \leftarrow s \cup inSubTbl$
  for each neighbor $n$ do
    if $a \in inAdvTbl$ such that $a.sender = n$ and $a$ matches $s$ then
      if $(s', m) \notin outSubTbl$ such that $m = n$ and $s'$ covers $s$ then
        Send $s$ to $n$
```

Broker Overlay

The performance of a pub/sub system is influenced by the organization of the brokers in the pub/sub network. There are several measures that characterize the quality of broker network:

Number and Placement of Brokers: Placement of brokers can have a huge impact on the system's performance. For instance, if the broker modules reside on the DSRC access points themselves, then a same amount of physical mobility would result in higher network mobility as compared to the case when a set of access points are connected to a broker node.

Network Diameter: Network diameter is the longest path between any two brokers in the network. It is important to minimize the number of hops traveled by any message. This will lead to reduced latency and processing load on the brokers.

Broker Topology: We chose to experiment with these two topologies in TRAC:

1. The TREE topology: For this network, a geographical area would be divided into a number of local regions. All the access points (LWAs) in a local region would be connected to one broker which resides on a separate node. These brokers then form the leaf nodes of a tree topology of brokers. The TREE topology provides an intuitive way for opportunistic aggregation and high overlapping of the multicast trees. This is beneficial for a system like TRAC that exhibits good locality properties. To aid new access points joining the network, a registry server would maintain a list of brokers serving each area. A new access point would contact the registry server and obtain the address of the broker responsible for its region thus initiating the process of joining the network.

2. The PEER topology: In this network, both the broker module and the LWA would reside on the access point. The access points are connected in a peer-acyclic topology i.e. the configuration of connections among the brokers form an acyclic graph. Such configuration might provide a better load balancing when compared to the TREE topology. However, since the hierarchical aspect of the network is lost, locality would suffer. For new access points, a distributed database (like DNS) as proposed in [8] can be used. The database will be hierarchically distributed by region and will maintain a registry of access points and their GPS co-ordinates. Whenever a new access point wants to join the network it registers itself with the database and obtains a list of neighboring access points. The new access point would then connect only to its nearest neighbor. An acyclic network can be thus be created in an incremental fashion. Also, this would preserve the locality properties in TRAC.

Design Aspects

We will now discuss the various design choices we made in TRAC:

Pub/Sub Representation Languages: TRAC uses a simple subscription language where publications are represented as $(attr, val)$ pairs and subscriptions as predicates. This falls under the FAST class of algorithms defined in [14]. These algorithms have a relatively efficient matching algorithms that can run in O(1) time [7], [4]. The simple language is limited in expressiveness when compared to the COMPLEX class of algorithms [17], [6]. However, COMPLEX class of algorithms require heavier routing computations that has adverse effects on the system when mobility is introduced [14]. Since TRAC operates in a highly mobile environment we chose to use a simple subscription language. Moreover, the expressiveness of the simple subscription language is adequate for the purposes of traffic information dissemination.

Advertisements and Publications: Each publication has the attributes: seg_id (road segment id), $speed$ and ts (timestamp of publication). Hence, the publications would be of the form $\{(seg\_id = r_i, speed = s_i, ts = ts_i), (seg\_id = r_j, speed = s_j, ts = ts_j), \ldots, (seg\_id = r_n, speed = s_n, ts = ts_n)\}$. An advertisement is used to define the set of publications potentially generated by a publisher. Formally, an advertisement is expressed as a disjunction of set of boolean predicates. A predicate is expressed as $(attr\ rel\_op\ val)$. Advertisements in TRAC would be of the form $\{seg\_id = r_i \vee seg\_id = r_j \vee seg\_id = r_n\}$, that is basically a list of road segments for which a publisher would send out the publications.

Subscription Model: TRAC provides users (subscribers) with two types of queries. (1) A user might want to know about the average speed on a set of roads of interest to him. For such queries, the subscription could simply be of the form $\{seg\_id = r_k \vee seg\_id = r_m \ldots \vee seg\_id = r_q\}$. Here each predicate is simply concerned with the seg_id i.e. there are no constraints on the average speed on that segment or the timestamp at which the data was published. Note the disjunction operator in the subscription, which simply states that the publications must match *atleast one* of the predicates of the subscription i.e. a publication should be delivered only if it carries information about atleast one of the road segments the user is interested in. (2) A user might not want to be continuously updated about the average speeds on the roads, but might only be interested when there is a congestion (the notion of congestion might vary with users). For such cases, users can have customized traffic alerts by registering a subscription of the form $\{(seg\_id = r_i \wedge speed \leq s_i \wedge ts \leq t) \vee (seg\_id = r_j \wedge speed \leq s_j \wedge ts \leq t) \ldots \vee (seg\_id = r_n \wedge speed \leq s_n \wedge ts \leq t)\}$. The use of conjunction operator implies that users would be notified with a publications only if it satisfies constraints on all the predicates.

Content Routing Algorithm: Typically pub/sub systems propose flooding of advertisements as the number of publishers are far less when compared to subscribers. Another approach would be to use subscription flooding (or directed flooding) as in diffusion routing where the number of subscribers are less when compared to the sources (publishers). However, in our problem scenario, publishers and subscribers are equal in number. Both the above approaches would cause excessive tree (de) constructions due to high mobility of publishers and subscribers. We therefore use the virtual publisher concept to completely decouple the mobility of the publishers. This indirection provided by the system then reduces the problem to the case of static publishers and mobile subscribers. Clearly, flooding of advertisements would be better in this case. We therefore chose to use the standard routing algorithm with the advertisements being flooded by the virtual publishers and subscriptions are propagated in the reverse path. The publications from the virtual publishers then follow the multicast tree.

Subscriber Mobility Handling Schemes

The concept of virtual publisher greatly reduces the (un) advertisement costs as the system is decoupled of publisher mobility. However, costs due to subscriber mobility cannot be avoided as the system has to store state per subscriber in order to deliver the publications. A parallel idea of *virtual subscriber* might be helpful, which would essentially mean that information about certain road segments would always be coming in to that broker irrespective of whether there are any actual subscribers. This would be useful in the case where certain road segments are popular. However, if these segments are popular enough, then the active subscriptions on the broker would cover the new subscriptions handling the virtual subscriber case implicitly. Hence, in order to further mitigate the effects of subscriber mobility, we propose the following schemes:

STANDARD**:** The algorithms involved in the STANDARD scheme are similar to the ones proposed in [14] except that (1) brokers interact only with the LWAs (2) brokers act as virtual publishers (3) vehicles only perform the publish and subscribe operations and are not involved in advertisements. Consequently, multicast trees are constructed per data and not per publisher. When a subscriber issues a MOVEOUT, the LWA issues the unsubscriptions to the broker module and clears the subscriber's state information subscription and connection tables. The broker on receiving the unsubscription forwards the unsubscription to all the neighboring brokers to which it had previously forwarded the corresponding subscription. Additionally, it finds out all the subscriptions which were previously not sent out as they were covered by this subscription. For each of these subscriptions, it finds out the matching advertisements and forwards the subscription. The standard scheme is depicted in Figure 3.

UNSUB-DELAY**:** The basic idea here is to reuse pre-existing multicast trees by exploiting the locality properties in TRAC. In this scheme, when an LWA receives an unsubscription message from a subscriber, it delays forwarding of this unsubscription to the broker for some amount of time. When the subscriber issues the subscription again (after connecting to one of the nearby access points), there might be a significant overlap between the new tree and the old tree which results in less message cost and latency.

PRESUB-NBR**:** In TRAC, there is a high possibility that a subscriber after moving out of an access point connects to one of the access points in the neighboring area. In the PRESUB-NBR scheme, an LWA exploits this property by sending the pre-subscriptions to all the neighboring access points (LWAs) within some radius. When the subscriber issues a subscription from one of these access points, as the multicast tree would already been built, the subscriber can directly receive the publications. The latency as perceived by the user would be minimal in this case. If a subscriber does not connect to a pre-subscribed access point, unsubscriptions are issued after a timeout.

PRESUB-PRED**:** The PRESUB-NBR scheme results in a higher message cost as pre-subscriptions are sent to all the neighboring access points. In the PRESUB-PRED scheme, an LWA upon receiving an unsubscription predicts the next neighboring access point based on the road segments in the subscribed path and sends out the pre-subscription. Even though the PRESUB-PRED scheme requires more processing on the part of LWA, it results in lesser message overhead.

III. EVALUATION

In this section, we first present the metrics for evaluation and then the simulation set up is described followed by the simulation results.

Evaluation Metrics

Message cost: This is the total number of messages exchanged between the brokers for constructing/tearing down the advertisement/multicast trees.

Average delivery ratio: We define this as the ratio of actual number of road segments for which the publications were delivered to the number of road segments in subscribed path (i.e. the subscription length).

Average tree building time: This is the time required for the subscriptions to reach all the potential sources (brokers). The tree building time begins when a subscription message is sent to the access point (LWA) by a subscriber. And it is complete when the subscription messages stop propagating in the network and the multicast tree is built.

Average latency: Latency is defined as the minimum time required for the subscription messages to reach at least one source (broker) for each road segment present in the subscription.

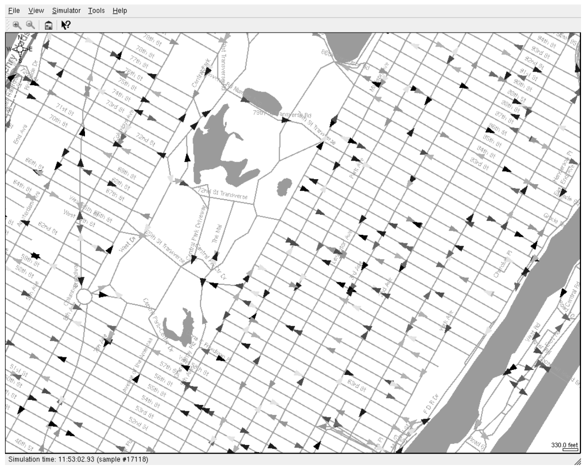
File View Simulator Tools Help
Simulation time: 11:53:02.93 (sample #17118)
330.0 feet

Fig. 4. TRAC Simulator with 650 vehicles driving around in Manhattan, NY, USA

Diffusion delay: Diffusion delay refers to the amount of time required for a publication to be delivered to all the interested subscribers i.e. the diffusion time begins when the publisher sends out the publications and it ends when these publications stop propagating in the network.

Average Cost per Subscription: This is the ratio of total number of subscription/unsubscription messages to the total number of subscriptions issued by the users.

TRAC Simulator

In order to accurately evaluate any vehicular traffic monitoring architecture, it is important to generate realistic inputs to the system. For this purpose, we have developed our own TRAC simulator which is based on GrooveSim [12]. GrooveSim generates a graph based abstraction of streets for any place in the USA by importing the TIGER/Line (Topologically Integrated Geographic Encoding and Referencing) [2] files. The TIGER/Line files provide a digital database in which road segments are represented by records including their location in latitude and longitude, name, address ranges and speed limits. Vehicles can be placed at random addresses in the imported map and then driven on the roads based on different trip/speed models. Vehicular movements would then be more realistic as TIGER/Line maps restrict the trajectories to the actual roadways. We developed the TRAC Simulator by enhancing GrooveSim and integrating it with a discrete event based simulator which modeled a publish/subscribe system. Communication between the vehicles, access points and the brokers is modeled using simple message queues taking appropriate latencies into account.

Simulation Setup

We simulated our experiments using the map of Manhattan county, New York, USA (service area of 23.7 square miles with a total of 9312 road segments). A total of 92 APs (LWAs) were placed at road intersections, uniformly distributed with a mean distance of around 600 meters. For the TREE topology, the service area was divided into 16 regions, each assigned to

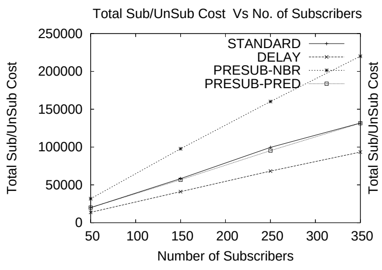

Fig. 5. Message Cost

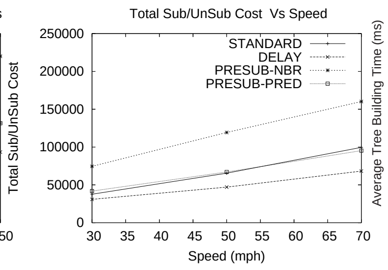

Fig. 6. Message Cost

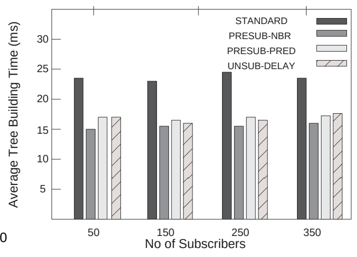

Fig. 7. Avg. Tree Building Time

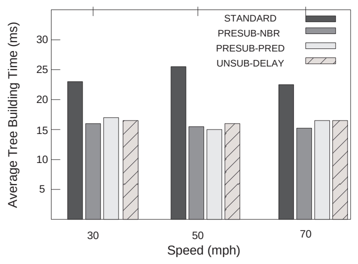

Fig. 8. Avg. Tree Building Time

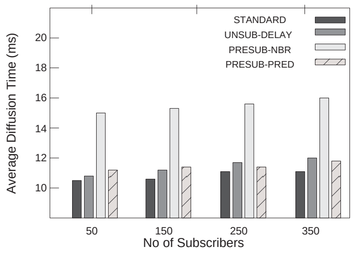

Fig. 9. Average Diffusion Time

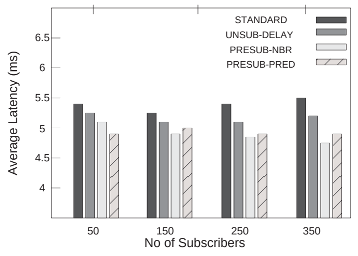

Fig. 10. Average Latency

Mobility Scheme	Cost per Subscription
STANDARD	1
PRESUB-NBR	0.71
PRESUB-PRED	0.43
UNSUB-DELAY	0.52

Fig. 11. Avg. Cost per Subscription (Normalized wrt. STANDARD)

Mobility Scheme	Cost per Subscription
STANDARD-TREE	1
STANDARD-PEER	2.42
PRESUB-NBR-PEER	1.73
PRESUB-PRED-PEER	1.26
UNSUB-DELAY-PEER	1.57

Fig. 12. Avg. Cost per Subscription (Normalized wrt. STANDARD-TREE)

a broker. These 16 brokers were leaves of a tree of height 4 and degree 4, resulting a total of 21 brokers. For peer topology, the access points were added in an incremental fashion with each new access point connecting only to its nearest neighbor to form an acyclic graph. The effective range of access points was reduced to around 100-125 meters, based on results in [16], for the publish/subscribe operations.

We initialized 650 vehicles at random locations in the area. The velocities of the vehicles were varied from 30mph to 70mph. Fixed speed model was used for our simulation so as to better understand the effects of mobility on TRAC. Each vehicle recorded its position (GPS coordinates) producing the $(seg\_id, speed, ts)$ tuples at a maximum rate of 5Hz[4]. The vehicles subscribed to a randomly chosen path whenever under the coverage of an access point and moved along the subscribed path until they reached the destination. When a vehicle moved out of an access point's range, MOVEOUT and unsubscription requests were issued to the corresponding LWA.

The latency for a vehicle's MOVEIN operation was set to 250ms and that of communication links was set to 2ms. These values were taken from [18], [14]. We also took routing computations into account. Since our subscription language is simple, a FAST matching algorithm can be used to perform the routing computations. In accordance with the results in [7], [14] we assumed a constant time of 2ms for the operations of covering, intersection, and matching. Hence, the total publication time was 2ms (matching), subscription and advertisement processing times were 4ms (covering+intersection). The brokers send out the new publications every 5 seconds. For PRESUB-NBR approach, pre-subscriptions were sent to all the neighboring access points inside the range of 1000 meters. For PRESUB-PRED approach, the next access point was figured out based on the subscribed path. The expiration timer for pre-subscriptions was set to 60 seconds and that for the cached data was set to be 3 minutes. A 50 second delay timer is used for UNSUB-DELAY scheme. The simulation time was set to be 30 mins. We also had an initial warm up phase of 15 mins in which vehicles only published the data (subscriptions were disabled). This was done in order for the system to reach a steady state, obtaining information for almost all the roads segments in the area.

In these experiments, we evaluate the costs involved in supporting (1) mobility and (2) increasing number of subscribers. Since, the vehicles choose a random path, subscribe only when they come under the range of an access point and unsubscribe as soon as they reach their destinations, the total number of vehicles covered by an access point at any point of time would be less than the actual number of vehicles served by the system. In our simulations, the maximum number of subscribers at any point of time was found to be 350. Unless otherwise stated all vehicles move at a speed of 70mph, there are 250 subscribers that are subscribed to a path of 15 road segments and a TREE topology is used for the broker network.

Results

Message Cost: Figure 5 shows the plot of the total cost incurred due to subscription/unsubscription messages. As the number of subscribers increase, the *aggregate* mobility increases and so does the cost of supporting it. PRESUB-NBR scheme requires more messages than any of the schemes as the pre-subscription messages are sent to all the neighboring access points. The PRESUB-PRED scheme performs well with its message cost almost equal to that of STANDARD scheme as

[4]This is the maximum rate supported by the latest GPS receivers

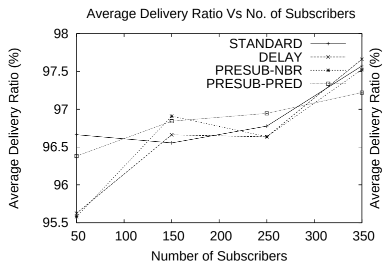

Fig. 13. Avg. Delivery Ratio Without Cache

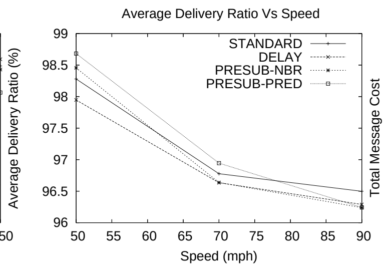

Fig. 14. Avg. Delivery Ratio with Cache

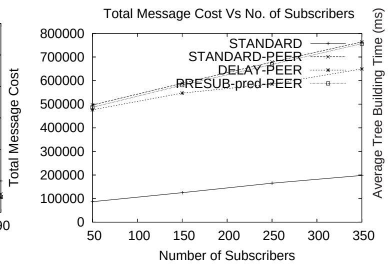

Fig. 15. Message Cost

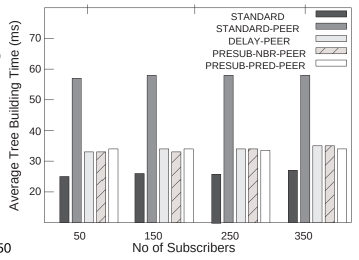

Fig. 16. Average Tree Building Time

the pre-subscription messages are sent to only one neighboring access point. The UNSUB-DELAY scheme performs better than the other schemes as no extra messages are sent. Moreover, as most of the times the subscriber moves in to one of the neighboring access points connected to the same broker, the subscription costs are further reduced when compared to the STANDARD scheme. Figure 6 shows the number of subscription/unsubscription messages with increasing mobility for 250 subscribers. A faster speed results in more frequent tree building and hence a higher message cost. As with the previous case PRESUB-NBR has increased message cost and UNSUB-DELAY scheme performs the best.

Average Tree Building Time: Figure 7 shows the average time required to rebuild the trees against increasing number of subscribers. The STANDARD scheme performs very well with a tree building time of less than 25ms. The PRESUB-NBR and PRESUB-PRED schemes further reduce the tree building time as the subscriber tree is already built before the actual subscriptions are sent. UNSUB-DELAY scheme also performs well as in most of the cases, the subscriber connects to the same broker (but a different AP). It is important to note that all the protocols scale well with increasing the number of subscribers. Even when routing computations are taken into account, there was no increase in the average tree building time. This is because TRAC uses a simple subscription language where a FAST matching algorithm is used. Figure 8 shows that tree building time is hardly affected by mobility. Again, the proposed mobility schemes perform much better than the STANDARD scheme.

Average Diffusion Time and Latency: In Figure 9 the average diffusion time is plotted against the number of subscribers. As the number of subscribers increase, it takes longer for a publication to reach all the interested subscribers and hence the diffusion time increases. Here, the STANDARD scheme requires a lesser diffusion time as the number of actual subscriptions at any point of time is higher for the proposed mobility schemes. The diffusion time for PRESUB-NBR is higher as a publication has to reach more number of subscribers compared to the other schemes.

Fig. 10 shows the the average latency with increasing number of subscribers. The trend is similar to that of average tree building time with the mobility schemes performing better than the STANDARD scheme. However, similar to the diffusion time, the latency does not vary significantly with the number of subscribers thus proving that our architecture is scalable.

Average Cost Per Subscription: The average cost for STANDARD scheme was 7.24 messages/subscription. Figure 11 shows the normalized cost. The mobility schemes incur lesser cost than STANDARD scheme due to the increased covering by delayed and prefetched subscriptions. PRESUB-PRED performs better than any other scheme at the cost of extra processing at the LWA.

Average Delivery Ratio: Figure 13 shows the average delivery ratio for increasing number of subscribers when caching is not used. At a high mobility (70mph), the average delivery ratio was found be at around 97%. Since publications are sent out every 5 seconds, the delivery ratio suffers in some of the cases where a subscriber's MOVEIN and MOVEOUT occur within the 5 second interval. Further, as the number of subscribers increase, covering of subscriptions increases the delivery ratio. With an increase in the speed of the subscribers, the average delivery ratio decreases as shown in Figure 14. However, even at 70mph, the delivery ratio was found to be around 99.5% when the publications are cached for a short duration (we purge the cache after 3 mins in our simulation) at the access points.

Subscription Length: We ran experiments to measure the effect of subscription length for STANDARD scheme. A higher subscription length would have to reach out more number of publishers thus resulting in increased latency and number of hops to be traversed. We found that the average latency increased from 4.2ms for a subscription length of 5 to 5.4ms for a subscription length of 20. Similarly, the average cost per subscription increased from 6.1 to 7.3. These results indicate that the effect of subscription length on the system is minimal.

Broker Topology: As discussed in section II, broker topology can have a major impact on the performance of any pub/sub system. Figure 15 shows the total message cost for STANDARD scheme when using TREE and PEER broker topologies. The STANDARD-PEER scheme incurs an increased message cost when compared to STANDARD-TREE scheme because (1) PEER topology has a larger network diameter (2) Increased locality in TREE topology results in more covering and decreased costs. Figure 16 shows the average tree building time. As expected, PRESUB-NBR-PEER and PRESUB-PRED-PEER schemes have a lesser tree building time when compared to STANDARD-PEER, but the STANDARD-TREE incurs even lesser cost. Again, this shows the importance of choosing the right broker topology when designing a pub/sub system. Our experiments with average latency for the above schemes also showed a similar trend. Average cost per subscription plotted against the above schemes in Figure 12 again shows that while the mobility schemes reduce the average number of hops traversed, the STANDARD-TREE scheme performs much better than the rest.

Summary

The results discussed in the previous subsection prove that our STANDARD scheme is scalable in terms of tree building time, latency, diffusion time and message cost. The proposed mobility schemes show an improvement over STANDARD for cost per subscription and provide a lower tree building time. However, they come at an added cost of more computation by the brokers and in some cases (for eg. PRESUB-NBR) a higher message cost. A tree broker topology is shown to perform better and whenever possible brokers should be configured using such a hierarchy.

IV. RELATED WORK

There has been a a lot of interesting prior work in the area of mobile publish/subscribe systems. Some studies have also focused on such systems involving vehicles as participants. The key difference between TRAC and the prior work is that TRAC defines the concept of virtualization to build a system supporting real-time efficient delivery of data in a very dynamic environment.

Muthusamy et. al. presented an extensive study and designed protocols to handle mobile publishers in [14]. They define optimizations to handle publisher mobility by reducing the cost and time to rebuild the advertisement and multicast trees. TRAC does not build trees for every publisher. TRAC has virtual publishers for semantically-unique pieces of information. TRAC also deals with mobile subscribers that are not dealt with in [14].

In [11], the authors have explored the design of a system that can be used for traffic monitoring. However, the study does not present a detailed description of how the data is being routed through the network to from the mobile publishers to the mobile subscribers.

An extensive study on the use of 802.11b within vehicles is done as part of the Drive-thru Internet project [16]. The authors have suggested the idea of a traffic monitoring application without going into the details of designing such an application.

[13], [9], [15] have all proposed the use of vehicular adhoc networks to propagate information amongst various entities. We believe that the use of adhoc networks limits the scalability of the system as well as localizes information to a large extent.

V. CONCLUSION

In this paper, we introduced TRAC: a scalable, distributed traffic monitoring and data dissemination system. TRAC is based on a publish/subscribe data delivery model. TRAC enables high speed vehicular users to become both publishers and subscribers and provides a mechanism for delivering location-aware information to the vehicular users in real-time. In the design of TRAC, we propose the concept of virtual publisher to completely decouple publisher mobility from the system and enable traffic information aggregation from multiple publishers. The locality properties exhibited in TRAC mitigate the effects of subscriber mobility to good extent. We proposed some mobility schemes that further exploit these properties by introducing pre-subscriptions and delayed-unsubscriptions.

The proposed architecture was evaluated through simulation experiments that accurately model vehicular movements within a real street map-based topography. Our simulations show that while the message load on the system increases linearly with increase in mobility, the average tree building time and latency are impervious. TRAC is able to achieve a high delivery ratio even when subscribers move at around 70mph. Further, the delivery ratio improves with the increase in the number of subscribers thus exhibiting good scalability properties. Finally, we also show the impact of broker topology on performance of TRAC.

REFERENCES

[1] Intelligent transport systems (its) america. *IVI Candidate User Services*, 2001.

[2] http://www.census.gov/geo/www/tiger. *TIGER/Line 2004. US Geological Survey (USGS) topographic maps*, 2005.

[3] I. Burcea, H.-A. Jacobsen, E. de Lara, V. Muthusamy, and M. Petrovic. Disconnected operation in publish/subscribe middleware. In *International Conference on Mobile Data Management*, 2004.

[4] A. Carzaniga and A. Wolf. Forwarding in a content-based network. In *ACM SIGCOMM*, 2003.

[5] G. Cugola, E. D. Nitto, and A. Fuggetta. The jedi event-based infrastructure and its application to the development of the opss wfms. In *IEEE Transactions on Software Engineering*, 2001.

[6] Y. Diao and M. Franklin. Query processing for high-volume xml message brokering. In *Tech. report, UC Berkeley*, 2003.

[7] F. Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *ACM SIGMOD*, 2001.

[8] G. Finn and J. Touch. Network construction and routing in geographic overlays pdf. In *ISI Technical Report TR-2002-564*, 2002.

[9] S. Ghandeharizadeh and B. Krishnamachari. C2p2:a peer-to-peer network for on-demand automobile information services. In *GLOBE*, 2004.

[10] Y. Huang and H. Garcia-Molina. Publish/subscribe in a mobile environment. In *International Workshop on Data Engineering for Wireless and Mobile Access*, 2001.

[11] L. Iftode, C. Borcea, N. Ravi, and T. Nadeem. Exploring the design and implementation of networked vehicular systems. In *DCS-TR-585, Rutgers University*.

[12] R. Mangharam, D. S. Weller, D. D. Stancil, R. Rajkumar, and J. S. Parikh. Groovesim: a topography-accurate simulator for geographic routing in vehicular networks. In *VANET*, pages 59–68, New York, NY, USA, 2005. ACM Press.

[13] A. Moukas, K. Chandrinos, and P. Maes. Trafficopter: A distributed collection system for traffic information. In *Proceedings of CIA*, 1998.

[14] V. Muthusamy, M. Petrovic, and H.-A. Jacobsen. Effects of routing computations in content-based routing networks with mobile data sources. In *MOBICOM*, 2005.

[15] T. Nadeem, S. Dashtinezhadd, C. Liao, and L. Iftode. Trafficview: Traffic data dissemination using car-to-car communication. In *ACM Sigmobile Mobile Computing and Communications Review*, 2004.

[16] J. Ott and D. Kutscher. Why seamless? towards exploiting wlan-based intermittent connectivity on the road. In *TERENA Networking Conference*, 2004.

[17] M. Petrovic, H. Liu, and H.-A. Jacobsen. G-topss: fast filtering of graph-based metadata. In *WWW*, pages 539–547, 2005.

[18] M. Shin, A. Mishra, and W. A. Arbaugh. Improving the latency of 802.11 hand-offs using neighbor graphs. In *ACM Mobisys*, 2004.